Contribution of the language network to the comprehension of Python programming code

Abstract

Does the perisylvian language network contribute to comprehension of programming languages, like Python? Univariate neuroimaging studies find high responses to code in fronto-parietal executive areas but not in fronto-temporal language areas, suggesting the language network does little. We used multivariate-pattern-analysis to test whether the language network encodes Python functions. Python programmers read functions while undergoing fMRI. A linear SVM decoded for-loops from if-conditionals based on activity in lateral temporal (LT) language cortex. In searchlight analysis, decoding accuracy was higher in LT language cortex than anywhere else. Follow up analysis showed that decoding was not driven by presence of different words across function, “for” vs “if,” but by compositional program properties. Finally, univariate responses to code peaked earlier in LT language-cortex than in the fronto-parietal network. We propose that the language system forms initial “surface meaning” representations of programs, which input to the reasoning network for processing of algorithms.

Introduction

The invention of computer programming and its applications (e.g., artificial intelligence) have altered human society and are fast becoming a central aspect of employment and education in the modern world. The cognitive and neural mechanisms that enable the human brain to support this important cultural skill are poorly understood. A key outstanding question is the degree to which the neurocognitive system that supports natural language processing is involved in understanding and producing programming code (Fedorenko, Ivanova, Dhamala, & Bers, 2019; Ivanova et al., 2020; Liu, Kim, Wilson, & Bedny, 2020; Peitek et al., 2018; Prat, Madhyastha, Mottarella, & Kuo, 2020; Siegmund et al., 2014).

Prima facie support for the idea that the language system is "recycled" for code comprehension comes from the fact that programming languages borrow some elements of natural language. Words like "for", "if", "and", "or", and "return" are used almost universally across programming languages, and their meanings are partly preserved. Even opaque and older function names (e.g., "chmod", "mkdir" in bash scripts) are abbreviations of English words rather than arbitrary letter combinations. The syntax of natural and programming languages also share features, such as hierarchical structure and recursion (Fitch, Hauser, & Chomsky, 2005). Natural languages are recursive because a phrase can be embedded within another phrase of the same syntactic category (Friederici, Chomsky, Berwick, Moro, & Bolhuis, 2017; Hauser, Chomsky, & Fitch, 2002; Yang, Crain, Berwick, Chomsky, & Bolhuis, 2017). Programming languages contain data structures, such as lists and trees, that can be recursive in the same way; furthermore, functions are allowed to call other functions, including themselves, as subroutines. Reading code can be thought of as partly analogous to reading natural language, where progressively more abstract and larger structures (e.g., functions) are constructed from lawful combinations of discrete symbols at lower levels of representation (i.e., letters, words) (Fedorenko et al., 2019). These parallels predicts overlap in the neural representation of natural and programming languages (Fedorenko et al., 2019; Fitch et al., 2005; Pandža, 2016; Peitek et al., 2018; Portnoff, 2018; Prat et al., 2020).

On the other hand, there are key differences between natural and programming languages. While English word forms do appear in computer code, the meanings of these symbols are not exactly the same as in natural languages. More generally, programming languages lack symbols that have the rich semantics of lexical items (e.g., "dog" or "walk"). Grammatical categories such as nouns, verbs, adjectives, and prepositions do not have clear counterparts in programming languages (e.g., an object in code can have both verb-like and adjective-like attributes). Furthermore, the rules for combining basic units are distinct: while human natural and programming languages both have function-argument structure, scope, and variable binding at the semantic level, only human natural languages have grammatical relations such as subject and direct object or information structure such as topic and focus. While natural languages are rife with ambiguity (i.e., a given sequence of words can have multiple lexical and syntactic parses with different meanings), the relation between code and meaning is deterministic. Programming languages may in fact have more in common with logical reasoning than with natural language. Code and formal logic both make use of conditions like "if", quantifiers like "for all", and logical operators such as "and", "or", and "not". In both systems, these expressions are interpreted

deterministically and without the pragmatic enrichment that is so characteristic of human natural language.

The available empirical evidence suggests that domain-general logical reasoning systems, rather than language networks, support programming (Dehaene, Al Roumi, Lakretz, Planton, & Sablé-Meyer, 2022; Fedorenko & Varley, 2016; Monti, Parsons, & Osherson, 2009, 2012). Studies of individual differences find correlations between programming learning outcome and logical, analogical, and deductive reasoning abilities (McCoy & Burton, 1988; Pea & Kurland, 1984; Prat et al., 2020; Shute, 1991). Recent functional magnetic resonance imaging (fMRI) studies have found activity in fronto-parietal reasoning areas, rather than perisylvian language circuits, when comparing programming tasks to various control conditions: code reading vs. reading algorithms written in plain English (Ivanova et al., 2020), code reading vs. syntactic bug finding (Siegmund et al., 2014), code reading vs. prose reading (Floyd, Santander, & Weimer, 2017), and searching for semantic bugs which prevent a program from implementing its intended algorithm vs. reading a bug-free program (Castelhano et al., 2018). The same fronto-parietal network is also involved in code writing as opposed to prose writing (Krueger et al., 2020), and even when participants were covertly crafting the program without actually typing it (Xu, Li, & Liu, 2021). In many of these studies, the control condition involved language (e.g., prose reading), leading to the concern that language-related activation was subtracted out. However, Liu et al. (2020), compared code comprehension to memorizing scrambled code, where the scrambled code lacked meaningful words and sentential structure. Nevertheless, the code vs. scrambled code contrast still identified fronto-parietal and not perisylvian language networks.

A possible conclusion from these data is that the language system plays little role in the processing of computer code (Ivanova et al., 2020; Liu et al., 2020). At the same time, it is not clear that the involvement of the language system in code processing can be entirely dismissed. One recent study found that while Python code and natural language do not activate the same cortical networks, they do show co-lateralization across individuals, suggesting some relationship between them (Liu et al., 2020). Moreover, examining univariate responses relative to control conditions (as reviewed above) is not the only way to test whether a neurocognitive system is involved in a particular task.

It remains possible that even though the fronto-temporal language system does not show large activity during code processing (perhaps because natural rather than programming languages are its preferred stimulus), multivariate patterns of activity in the language network still represent information relevant to programming code. Such dissociations between univariate and multivariate results have been observed in other domains of cognitive neuroscience. For example, despite low overall activity levels, early sensory areas contain information about the contents of visual working memory (Bettencourt & Xu, 2016; Emrich, Riggall, LaRocque, & Postle, 2013; Ester, Serences, & Awh, 2009; Harrison & Tong, 2009; Riggall & Postle, 2012).

Only a handful of studies have used multivariate methods to study programming and most of these have not looked at decoding in language regions. Ikutani et al. (2021) and Liu et al. (2020) found that different types of programming algorithms can be decoded based on the spatial activation pattern in the fronto-parietal reasoning network. Furthermore, decoding accuracy in the

fronto-parietal network correlated with behavioral performance on the behavioral task where participants sorted programming scripts based on the underlying algorithms (Ikutani et al., 2021). However, neither of these studies specifically looked at decoding in functionally localized language areas.

Recent evidence for the hypothesis that language areas may in fact contain multivariate information about programming code comes from Srikant et al. (2022). They found that a linear classifier trained on activity patterns from the language network could be used to distinguish between different types of control structures (FOR loop, IF conditional, or sequential operations lacking FOR and IF) and data types (string vs. numeric) in Python programming scripts. These results suggest that, contrary to the inferences from univariate measures, the language network does show sensitivity to the content of computer code.

In the current study, we sought to replicate and extend these results by probing in greater detail the contribution of the language network to program comprehension. We used data from a previous publication (Liu et al., 2020). In the previous work, we did not conduct any multi-variate pattern analysis (MVPA) in the language network. In our current study, MVPA was used to decode FOR from IF functions based on activity patterns within classic language regions identified in individual participants. While undergoing functional magnetic resonance imaging (fMRI) scans, expert Python programmers read short functions, each of which contained a single FOR loop or a single IF conditional. The same participants performed a language localizer task where sentence comprehension was compared to solving math equations. This functional localization allowed us to conduct our analyses within the neural population most sensitive to linguistic content within each individual participant.

The current experiment went beyond previous studies by asking whether the language network is sensitive to the compositional meaning of programming functions or is restricted to retrieving the meanings of programming keywords, such as “return”, “for” and “if”. If the language-networks’ role is restricted to retrieving word meanings, decoding might be based purely on the presence of distinct lexical items across different code function types. This hypothesis is consistent with the available data because in the only prior study to find decoding in the language network, the decoded functions contained different words, in addition to differing in line structure (Srikant et al., 2022).

A feature of the stimuli in the current study made it possible to test the hypothesis that the language network is sensitive to the compositional structure and not just the lexical items of code: the same participants read real Python code functions and memorized similar functions presented with all the words in scrambled order, i.e., “scrambled fake function”. Each scrambled “fake” function was generated from a real Python function by scrambling the words and symbols within each line. As a result, the lexical items within each scrambled function were identical to a real function, so words like “for” and “if” were preserved. However, the fake functions lacked the meaning and structure that is present in real Python functions. If decoding of IF vs. FOR functions in language regions is driven by the presence of different lexical items, we should be able to decode not only the real but also the fake functions. By contrast, if decoding in the language

network is driven by the meaning and/or structure of the Python code, then we should find decoding of real but not fake function.

Next, we compared the temporal dynamics of neural responses to Python functions across language and fronto-parietal reasoning networks to test whether these networks contribute to different aspects of code comprehension. We hypothesized that the language system is responsible for the initial meaning extraction from programming text, whereas fronto-parietal logical reasoning system subsequently creates a mental model of the programming algorithm. This latter mental model is a more in-depth representation than the surface meaning extracted by the language system. It is more flexible, containing variables that can take on specific values and be entered into functions. Based on this hypothesis, we predicted that the language system contributes to code comprehension earlier than the fronto-parietal reasoning system and that the blood-oxygen level dependent (BOLD) signal responses to code would peak earlier in language relative to fronto-parietal systems.

Methods

As this study consists of further analyses on the data collected for a previous publication, the participants, experiment design, and data acquisition procedures are identical to what was described previously by Liu et al. (2020). However, for the sake of completeness, we still briefly describe these aspects of the study.

Participants

Fifteen individuals participated in the study (three women, twelve men, age range 20–38, mean age = 27.4, SD = 5.0). Participants had an average of 5.7 years of Python programming experience (range: 3–9, SD = 1.8). Other than self-report, Python expertise was evaluated with two Python exercises administered outside the MRI scanner. In the first exercise, participants answered what will be the output of a one-line Python statement (e.g. for “print(‘3.14’.split(‘1’))”, participants should type “[‘3.’, ‘4’]”). In the second exercise, participants saw a Python program snippet with a blank, along with a sentence describing what the snippet should do when executed. Participants were required to complete the snippet to fulfill the specification (e.g., for the snippet “a = ‘abc’; print(_____(enumerate(a)))” and the specification “What should be filled in the blank if we want to print out a collection of tuples enclosed in square brackets, rather than something like <enumerate object at 0x000001888D2B2678>?”, participants should type “list”). The first exercise evaluated participants’ knowledge of basic Python syntaxes and built-in functions (participants’ responses M = 82.9%, SD = 6.9%, range: 70–96%), whereas the second exercise evaluated participants’ ability to use their programming knowledge to solve a problem (M = 64.6%, SD = 16.6%, range: 37.5–93.75%). For detailed descriptions of these exercises, please refer to Liu et al. (2020).

All participants had normal or corrected to normal vision and none had been diagnosed with cognitive or neurological disabilities. All participants gave informed consent according to procedures approved by the Johns Hopkins Medicine Institutional Review Board (IRB protocol

number: NA_00087983).

fMRI task design and stimuli

Participants took part in an fMRI Python code comprehension experiment and a second localizer experiment for language, logical reasoning, and symbolic math. Participants also performed a multi-source-interference-task (Bush & Shin, 2006) which is not relevant to the current paper, and will not be discussed further.

The code experiment consisted of Python code condition and a “fake” code memory control. Real code comprehension trials involved the sequential presentation of three elements: a Python function (24s), an input to the function (6s), and a proposed output (6s). Participants judged whether the proposed output was correct and indicated their response via a yes/no button press. Each Python function contained exactly one control structure, which was either a FOR loop or an IF conditional. There were two variants for FOR functions, and two variants for IF functions. The first variant of FOR functions implemented the FOR loop in the canonical way, where a FOR loop began with the keyword “for”, followed by actions to be taken in each iteration. The second variant of FOR functions contained a Python-specific expression called “list comprehension”, where the keyword “for” was placed in the middle of a loop definition, rather than the beginning. In the first variant of IF functions, a conditional statement began with the keyword “if”, followed by the action to be taken if a condition was met. In the second variant, the keyword “if” was not used. Instead, we multiplied the action by the condition such that if the condition evaluated to false, the product of the multiplication was 0, indicating no action was taken. Despite the existence of the variants, all functions consisted of exactly 5 lines of code with the same patterns of indentation, such that FOR and IF functions, regardless of the variants, were visually similar. All functions took a character string as input and performed string manipulations. As discussed in detail below, analyses focused on the function comprehension portion of the trial, prior to input presentation. Each trial was followed by a 5-second inter-trial interval. Please see Figure 1 for example stimuli.

Prior to the experiment, participants were told they were going to see functions that work with character strings called “input”. They also practiced with these types of stimuli, so they were well aware that in the scope of this experiment, “input” referred to the input argument of the functions, rather than a Python built-in function which happened to have the same name.

Fake code memory trials had a similar structure to the code trails: a scrambled Python “function” (24s), followed by a one-line “input” (6s) and a scrambled one-line “output” (6s). Participants were instructed to remember the text presented during the first two phases of the trail. They then judged whether the one-line scrambled “output” matched any of the lines presented during the previous phases (including both the scrambled function and the scrambled “input”). As with real code, analysis focused on the “function” portion of the trail.

Every scrambled function was generated from a real Python function by separately scrambling each line of the real function at the level of words and symbols. Therefore, the words, digits and operators present in real functions were preserved in scrambled functions, but none of the scrambled lines comprised an executable Python statement. Like the real IF and FOR functions,

the fake FOR functions contained the word “FOR”, and the IF scrambled functions did not contain the word “FOR”. We contrasted reading and comprehending Python code against an explicit working memory task as an attempt to emphasize only the process of understanding the algorithm, but not the working memory processes associated with the comprehension.

There were six task runs in this scan. During each run, participants saw 8 real FOR functions, 8 real IF functions, and 4 fake scrambled functions (48 FOR, 48 IF, and 24 fake across runs for each participant). In each of the 6 task runs, each participant only saw either the “real” or the “fake” version of the same function, but not both.

The localizer experiment included language comprehension, logical reasoning, and symbolic math tasks all following the same structure. On language trials, participant judged whether two sentences, one in active and one in passive voice, had the same meaning (e.g., “The child that the babysitter chased ate the apple” vs “The apple was eaten by the babysitter that the child chased”). On formal logic trials, participants judged whether two logical statements were consistent. That is, whether the two statements logically infer each other (e.g., “If either not X or not Y then Z” vs “If not Z then both X and Y”). On math trials, participant judged whether the variable X had the same value across two equations (e.g., “X minus twenty-three equals forty-two” vs “X minus fifty-one equals fourteen”). In each trial, one of the two “sentences” appeared first, with the other following 3 s later. Both statements stayed visible on the screen for 16 s. Participants indicated their true/false judgment by pressing one of two buttons. The experiment consisted of 6 runs, each containing 8 trials of each type (language/logic/math) and six rest periods, lasting 5 s each. In this study, we localized the perisylvian fronto-temporal language network using the language>math contrast, and the lateral fronto-parietal logical reasoning network using the logic>language contrast (Kanjlia, Lane, Feigenson, & Bedny, 2016; Monti et al., 2009, 2012). The logic task was adapted from Monti et al. (2009) and Monti et al. (2012); whereas the language task was adapted from Kanjlia et al. (2016), which was also derived from Monti et al. (2012).

fMRI data acquisition and preprocessing

All functional and structural MRI data were acquired at the F.M. Kirby Research Center of Functional Brain Imaging on a 3T Phillips Achieva Multix X-Series scanner. T1-weighted structural images were collected in 150 axial slices with 1 mm isotropic voxels using a magnetization-prepared rapid gradient-echo (MP-RAGE) sequence. Functional T2*-weighted BOLD scans were collected using a gradient echo planar imaging (EPI) sequence with the following parameters: 36 sequential ascending axial slices, repetition time (TR) = 2 seconds, echo time (TE) = 0.03 seconds, flip angle = 70°, field of view (FOV) matrix = 76 x 70, slice thickness = 2.5 mm, inter-slice gap = 0.5, slice-coverage FH = 107.5, voxel size = 2.4 x 2.4 x 3 mm, PE direction = L/R, first order shimming. Six dummy scans were collected at the beginning of each run but were not saved. We acquired the data in one code comprehension session (six runs) and one localizer session (6 runs of language/math/logic), with the acquisition parameters being identical for both sessions.

The stimuli in both sessions were presented with custom scripts written in PsychoPy3 (https://www.psychopy.org/ (Peirce et al., 2019)). The visual stimuli were presented on a rear

projection screen, cut to fit the scanner bore. The participant viewed the screen via a front-silvered, 45°inclined mirror attached to the top of the head coil. The stimuli were projected with an Epson PowerLite 7350 projector. The resolution of the projected image was 1600 × 1200.

Data were analyzed using Freesurfer, FSL, HCP workbench, and custom in-house software written in Python (Dale, Fischl, & Sereno, 1999; Glasser et al., 2013; Smith et al., 2004). Functional data were motion corrected, high-pass filtered (128 s), mapped to the cortical surface using Freesurfer, spatially smoothed on the surface (6 mm FWHM Gaussian kernel), and prewhitened to remove temporal autocorrelation. Covariates of no interest were included to account for confounds related to white matter, cerebral spinal fluid, and motion spikes.

Analysis

Whole-cortex searchlight multivariate pattern analysis (MVPA)

In this analysis, we asked: in the whole cortex, where were the FOR and IF programming functions most distinctly represented? To answer the question, we conducted MVPA decoding to distinguish FOR and IF functions using local spatial activation patterns in the "searchlight" associated with each vertex on the cortical surface.

To prepare data for decoding, we constructed a general linear model (GLM) where each real code function (48 FOR and 48 IF) and each scrambled control function (12 FOR and 12 IF) was entered as a separate predictor with 24s duration, modeling the function presentation phase. A support vector machine (SVM) classifier was then trained and tested on the spatial pattern of z-statistics associated with beta parameters estimated by the GLM. The SVM classifier was implemented in the Python toolbox Scikit-learn (Chang & Lin, 2011; Pedregosa et al., 2011). For each vertex in the brain, one linear SVM classifier (regularization parameter C = 5.0) was trained and tested on the spatial pattern in a "searchlight" surrounding the vertex. The searchlight associated with a vertex consisted of all the vertices within a circle of 8mm diameter (according to geodesic distance) centered at the vertex (Glasser et al., 2013; Kriegeskorte, Goebel, & Bandettini, 2006). Searchlights containing sub-cortical vertex were excluded. The regularization parameter C for SVM classifiers indicates how much misclassification of the training data is allowed, where a larger value means less misclassification. But increasing C value also increases the risk of overfitting the training data, leading to reduced decoding accuracy when applying the model to the testing data. The default value of C provided by Scikit-learn was 1, and we selected a larger value to impose a harder margin to better avoid misclassification. Empirically, a reasonable C value ranges from 1 to 10 (https://www.ibm.com/docs/en/spss-modeler/18.2.2?topic=node-svm-expert-options). We selected 5, which falls at the center of this range, and used this value consistently throughout the analysis.

To eliminate any difference in the overall signal strength across MRI scanning runs, data were normalized within each run (Lee & Kable, 2018; Stehr, Garcia, Pyles, & Grossman, 2023). In each run, in each vertex on the cortical surface, the mean and standard deviation were computed across trials and used for normalization such that the mean was set to 0 and

standard deviation to 1. To avoid the dependency between trials from the same run artificially inflating the decoding accuracy, we performed a 6-fold leave-one-run-out cross-validation (Etzel, Valchev, & Keysers, 2011; Mumford, Davis, & Poldrack, 2014; Valente, Castellanos, Hausfeld, De Martino, & Formisano, 2021). In each cross-validation fold, the classifier was trained on the data from 5 out of the 6 task runs and tested on the left-out run. The resulting 6 accuracy values were averaged to derive the observed accuracy for one participant in one searchlight. In each searchlight, we used a one-sample t-test to test the 15-participant group mean accuracy value (Fisher z-transformed) against chance of 50% (also Fisher z-transformed).

To control for family-wise error rate (FWER), we applied a cluster-based permutation correction (Elli, Lane, & Bedny, 2019; Musz, Loiotile, Chen, & Bedny, 2022; Regev, Honey, Simony, & Hasson, 2013; Schreiber & Krekelberg, 2013; Stelzer, Chen, & Turner, 2013; Su, Fonteneau, Marslen-Wilson, & Kriegeskorte, 2012) with a vertex-wise cluster-forming threshold of uncorrected $p<0.001$, and a cluster-wise FWER threshold of $p<0.05$ (Eklund, Knutsson, & Nichols, 2019; Eklund, Nichols, & Knutsson, 2016; Winkler, Ridgway, Webster, Smith, & Nichols, 2014). Specifically, we shuffled the condition labels (FOR and IF) 100 times. For each shuffle, we derived one null group-mean accuracy map. For the observed group-mean accuracy map and each of the null maps, we applied a cluster-forming threshold of $p<0.001$. For each cluster, we computed its strength-over-spread, which is the average distance between each vertex in the cluster with the peak cluster, weighted by the decoding accuracy value of the vertex. We recorded the maximum strength-over-spread value among the clusters in each null accuracy map to form a null distribution of 100 maximum strength-over-spread values. A cluster in the observed map passed the correction if its strength-over-spread value was greater than the 95th percentile of the null distribution.

ROI definition

Separate GLMs were constructed for the code experiment and localizer scans for the purpose of generating region of interest (ROI) search spaces and individual functional ROIs (fROIs). Specifically, our analysis focused on four ROIs: language-responsive lateral temporal cortex (LT), code-responsive intraparietal sulcus (IPS) and lateral prefrontal cortex (PFC), and a control region, the medial occipital primary visual cortex (OCC). Each ROI served as a search space within which we defined functional ROI (fROI) for each individual. The fROI approach was adopted to account for the known individual differences of neural populations engaged by a cognitive task (Nieto-Castañón & Fedorenko, 2012). It enabled us to select the subject-specific neural populations engaged by experiment conditions within a larger group-based ROI search space. How the group-based ROI search spaces and the fROIs within each individual were defined is described in detail below.

In the localizer GLM, one predictor was included for each of the three conditions (sentence, math, and logic), modelling the 16s duration when the pair of statements was visible. A separate predictor was entered to model the 5s rest period between trials. For further details see Liu et al., (2020). The group contrast of sentence>math ($p<0.05$, FWER cluster-corrected) was used to define the language-responsive lateral temporal cortex (LT) ROI search space, and the individual sentence>math contrasts were used to select language-responsive fROIs within the LT ROI search space. On the other hand, the group contrast of logic>sentence ($p<0.05$, FWER

cluster-corrected) was used to define the intraparietal sulcus (IPS) and the lateral prefrontal cortex (PFC) ROI search spaces.

For the coding experiment GLM, each condition (5 in total: 2 variants of FOR functions, 2 variants of IF functions, and scrambled fake function) were entered as a separate predictor to model the duration of function presentation (24sec). The individual real>fake function contrasts were used to select code-responsive fROIs within the IPS and the PFC ROI search spaces.

To generate the IPS, PFC (based on group logic>sentence contrast), and the LT (based on group sentence>math contrast) search-space masks, we combined the cortical parcels in the 400-parcel map reported by Schaefer et al.(2018) which included vertices activated in the contrast of interest. To define the OCC search space, we combined the peri-calcarine parcels from Schaefer et al.(2018) to form a search space covering the medial occipital primary visual cortex.

Within each ROI search spaces, we defined the fROI for each individual. Language-responsive individual subject fROIs were defined as vertices showing the strongest fixed-effect for the sentence>math contrast in the LT search space. For MVPA decoding, we selected the top 500 sentence>math vertices in LT. For percent signal change (PSC) analysis in LT, we selected the top 5% sentence>math vertices (Kanjlia et al., 2016; Kim, Kanjlia, Merabet, & Bedny, 2017). Specifically, within the LT search space, vertices with negative z-statistics in the sentence>math contrast were excluded, and the top 5% of vertices were selected from the subset of vertices with positive z-statistics (i.e., those preferring sentence over math). In the IPS and, PFC, individual code-responsive fROIs were defined in a similar fashion, but based on the real>fake function contrast from the code comprehension experiment. The average number of selected vertices are 130 in the LT, 70 in the IPS, and 101 in the PFC. For MVPA decoding, for each participant within each search space, we selected the top 500 vertices based on the fixed-effect real>fake function contrast. For the percent signal change (PSC) analysis in IPS and PFC, we used a leave-one-run-out approach (Glezer & Riesenhuber, 2013; Kriegeskorte, Simmons, Bellgowan, & Baker, 2009), taking data from 5 out of the 6 runs to select the top 5% of code-responsive vertices within a search space, and extracted the PSC time course from the held out run. For each participant, this process was repeated for all 6 runs and the results were averaged across folds. In the PSC analysis, for the code-responsive fROIs, the leave-one-run-out approach was adopted to avoid circular analysis where we extract neural responses during code reading from the fROIs defined by the responses during code reading. However, for the language-responsive fROIs, because we extracted the neural responses during code reading from the fROIs defined by an orthogonal localizer contrast, leave-one-run-out approach was neither necessary nor preferable.

ROI-based MVPA

In this analysis, we investigated the neural representations of algorithms in four ROIs: language-responsive LT; the code-responsive IPS and the PFC; and the control region OCC, which was not expected to encode information relevant to programming algorithms.

In this analysis, we used the same configuration for the SVM classifier, training data, and normalization scheme, and leave-one-run-out cross-validation as in the searchlight MVPA.

To determine whether decoding across IF vs. FOR functions was driven by the presence of different “lexical” items such as the word “for”, or rather by the compositional structure of programming functions, we trained and tested the same SVM classifier on scrambled fake functions. Recall that each fake function was derived from one real function, where words such as “for” or “if” were retained. As a result, fake functions can be divided into “for” fake functions and “if” fake functions. Except for the training and testing data, the procedures for fake function decoding was the same as real function decoding. Because participants saw four times as many real functions as fake functions, we conducted a separate control decoding analysis, where we trained the same SVM classifier on only a quarter of the real function data.

Percent signal change (PSC) analysis

We examined the time courses of percent signal change (PSC) of the blood-oxygen level dependent (BOLD) signal to study the dynamics of the neural responses to programming functions in each fROI. Specifically, we compared the (1) peak-to-trough amplitude, (2) peak time, and (3) average PSC of the time courses between the language-responsive LT and the code-responsive IPS and PFC.

PSC was calculated as [(Signal condition - Signal baseline)/Signal baseline], where baseline is the activity during the inter-trial interval. We extracted PSC from the duration of function presentation (FOR, IF, or fake), excluding activity related to the derivation of specific output or response processes. As introduced in the section regarding ROI definition, in the language-responsive LT, for each participant and each run, we extracted PSC from the top 5% of active vertices in the fixed-effect sentence>math contrast. The 6 resultant PSC curves were then averaged. In the IPS and PFC, we used a leave-one-run-out approach to select the top 5% of active vertices in the fixed-effect real>fake function contrast, and extract the PSC using the left-out run. This was repeated across a 6 runs, and the 6 resultant PSC curves were then averaged.

Since IPS and PFC showed similar time courses, they were averaged together. Therefore, the extraction resulted in one PSC time course per condition (FOR, IF, and fake) per participant, in either the language-responsive LT or the code-responsive lateral fronto-parietal network (IPS&PFC). The peak-to-trough amplitude was defined as the difference between the maximum and minimum value of a time course. The peak time was the time point when the maximum value happened. The average PSC of the time courses was computed by averaging the middle 14s of the time course (that is, the beginning and the last 5 seconds were not considered). Paired t-tests were conducted to compare these variables of interest.

Results

MVPA decoding of FOR and IF functions in language-responsive lateral temporal cortex

In whole-cortex searchlight analysis, we found reliable above-chance decoding of FOR and IF functions in a left lateralized fronto-temporal network, with the highest accuracy in the lateral superior temporal (BA 22, 39) and temporo-parietal cortices (the angular and the supramarginal gyri, BA 40). Smaller clusters were also found in left prefrontal cortex, left intraparietal sulcus and right temporo-parietal cortices (Figure 2).

Next, we used individual-subject ROI analysis to test whether in individual participants, language-responsive lateral temporal cortices contain neural populations that distinguish between IF vs. FOR Python functions. Lateral temporal ROIs chosen in individual participants for their language selectivity in the localizer scan (sentence>math) showed above chance classification of FOR and IF code functions (LT accuracy = 66.9%, Wilcoxon signed rank test against chance: $z=-3.41$, $p<0.001$). Decoding in language-responsive lateral temporal cortex was as high as decoding in IPS (accuracy = 67.8%, $z=-3.41$, $p<0.001$) and PFC (accuracy = 64.4%, $z=-3.41$, $p<0.001$), where vertices were chosen for their high responses to real over fake code. Wilcoxon signed rank tests showed that decoding accuracy values in the LT, the IPS, and the PFC, were higher than in an occipital cortex control region (OCC accuracy = 55.5%, $z=-2.54$, $p<0.05$; LT vs OCC: $z=-3.10$, $p<0.01$; IPS vs OCC: $z=-3.04$, $p<0.01$; PFC vs OCC: $z=-2.56$, $p<0.05$) (Figure 3). In sum, lateral temporal language-responsive areas showed high decoding accuracy for IF vs. FOR functions. Decoding in language-responsive LT was as high as or higher than in IPS and PFC areas identified for their univariate responses to code (Supplementary Figure 1).

One possibility is that language-responsive cortex is sensitive only to the “lexical” items present in code and not to compositional structure of code functions (e.g., presence of particular key words, such as “if” and “for”). Alternatively, language areas may be sensitive to the compositional structure of Python code functions. To distinguish between these possibilities, we conducted the same decoding using the neural responses to scrambled fake functions. Recall that in our experiment, each fake function contained all the words and symbols of the corresponding real function in a scrambled order, line by line. Therefore, like real FOR functions, all fake FOR functions contained the word “for”. Contrary to the lexical-only hypothesis, we did not find significantly above-chance decoding accuracy for fake code in language-responsive lateral temporal cortex, or in any of the other ROIs (LT: accuracy=46.7%, $z=-1.15$, $p=0.24$; IPS: accuracy=46.9%, $z=-1.43$, $p=0.15$; PFC: accuracy=49.2%, $z=-0.45$, $p=0.65$; OCC: accuracy=44.4%, $z=-1.92$, $p=0.053$). In the language-responsive LT and the code-responsive IPS and PFC, the decoding accuracy for real functions was significantly higher than for fake functions. Curiously, this difference was observed in the OCC (LT: $z=-3.01$, $p<0.01$; IPS: $z=-2.78$, $p<0.01$; PFC: $z=-2.22$, $p<0.05$; OCC: $z=-3.04$, $p<0.01$) (Figure 3). To account for the fact that there were four times as many real as fake functions in the experiment, we repeated the decoding analysis on one quarter of the real code data. For real code, decoding in the language-responsive lateral

temporal cortex, the IPS, and the PFC remained significantly above chance (LT: accuracy=65.8%, z=-3.08, p<0.01; IPS: accuracy=58.6%, z=-3.06, p<0.01; PFC: accuracy=60.3%, z=-2.93 p<0.01). In the LT and the IPS, the difference between the decoding accuracy for a quarter of real functions and for fake functions remained significant (LT: z=-3.01, p<0.01; IPS: z=-2.13, p<0.05) and marginally significant in the PFC (z=-1.88, p=0.06). In the OCC, neither the decoding accuracy for a quarter of real function or the difference between a quarter of real function and fake function reached significance (OCC: accuracy=51.1%, z=-0.25, p=0.80; difference with fake function decoding z=-1.61, p=11) (Supplementary Figure 1). This result suggests that multivariate patterns in language-responsive lateral temporal cortex (or IPS and PFC) cannot be explained by the lexical meanings of the particular words used in the functions but is related at least in part to the compositional structure of code.

Language vertices had weaker but earlier univariate responses to programming functions

In a sensitive individual-subject ROI analysis focusing on the top 5% of language responsive vertices in lateral temporal cortex (language>math), we observed a small but significant response to real over fake code in language-responsive LT (real functions: 0.095%, SD=0.26%; fake functions: -0.028%, SD=0.27%; t(14)=2.87, p<0.05). Consistent with previously reported whole-cortex results (Liu et al., 2020), univariate responses to Python code in LT were much smaller than in fronto-parietal cortices IPS&PFC (real functions: 0.65%, SD=0.27%; fake functions: 0.023%, SD=0.22%; real vs. fake t(14)=9.52, p<0.001), both compared to the memory control condition and compared to rest (FOR: mean PSC difference = 0.42% in the LT, SD=0.14%; mean = 1.1% in the IPS&PFC, SD=0.41%; paired t-test between ROIs t(14) = -6.10, p<0.001. IF: mean PSC difference = 0.53% in the LT, SD=0.20%; mean = 1.24% in the IPS&PFC, SD=0.42%; t(14) = -6.22, p<0.001).

Based on the hypothesis that language regions are involved in the representation of the initial “gist” of programming functions, we predicted that the neural response to programming functions should peak earlier in language-responsive lateral temporal cortex than in the fronto-parietal reasoning network (IPS and PFC). Consistent with this hypothesis, we observed significantly earlier signal peaks in language-responsive LT than the code-responsive fronto-parietal network (FOR: mean peak time = 8.87s in the LT, SD=6.39s; mean = 15s in the IPS&PFC, SD=3.72s; t(14)=-4.77, p<0.001. IF: mean peak time = 11.67s in the LT, SD=6.14s; mean = 15.93s in the IPS&PFC, SD=3.99s; t(14)=-4.07, p<0.005.) This result supports the hypothesis that language areas are involved in code comprehension than fronto-parietal systems (Figure 4). Separately comparing LT with IPS and with PFC led to the same results. The responses in either the IPS or the PFC were stronger and faster than in the LT (Supplementary Figure 2).

Discussion

We find that language-responsive lateral temporal cortices are sensitive to the contents of the programming functions. Linear classifier trained on activity patterns in the language network can

distinguish between FOR loops and IF conditional functions. Indeed, decoding accuracy in language-responsive lateral temporal cortex was as high as or higher than anywhere else on the cortical surface, including fronto-parietal networks that show strong univariate responses to code. This decoding result is consistent with a recent report by Srikant et al. (2022), which found above-chance decoding accuracy of control structures (FOR, IF, or a sequential script with neither FOR or IF) and data types in both the fronto-temporal language network and the fronto-parietal reasoning network. Interestingly, high decoding is found in the language network despite relatively low univariate activation to Python code in language areas.

The current results also suggest that language regions, as well as the fronto-parietal network, are sensitive to the compositional structure of code. Decoding in language regions is not likely to be driven solely by the presence of different lexical items across different Python functions since the scrambled "fake" control functions contained the same lexical items as individual real Python functions but did not show above chance decoding in language-responsive lateral temporal cortex (or in the IPS and PFC).

One caveat to our conclusion is that differences between decoding for real and fake might be partly driven by task demands, and not only the stimuli themselves. For the Python code, participants read for meaning, but for fake functions, they performed memorization. However, even in the memory control task, participants needed to attend to individual lexical items in the fake functions to correctly perform the task, making it unlikely that they were ignoring the lexical items. It is also possible that the unstructured nature of fake code stimuli contributed to the disengagement of the language network. That is, when the lexical items in a programming function are scrambled to disrupt the structure, they don't engage the language network as an actual function does. Base on the current data alone, we cannot rule out the possibility that lower decoding accuracy for fake as compared to real code is related in part to different task demands across these conditions or the greater variability of the position of the words in the fake functions.

Prior studies provide further evidence for the idea that language regions are not highly sensitive to lexical information in code (Srikant et al., (2022). Srikant et al. (2022) failed to decode between functions that contained variables in English (e.g., the variable holding the mean of two values was named "mean") as opposed to functions that contained the same variables named in Japanese spelled out with English alphabet (e.g., the variable holding the mean was named "heikin", which means "mean" in Japanese). Together, these findings suggested that the lateral temporal language areas (and fronto-parietal areas) are sensitive to the composition content of code functions, beyond the lexical items in a Python function.

Support for the idea that the language network plays some role in code comprehension comes from previous evidence of co-lateralization. In a previous publication on the same dataset, we found that fronto-parietal responses to Python code are left-lateralized and importantly co-lateralized with the language network across individuals (Liu et al., 2020). Notably, since the current results come from the same dataset as that prior analysis, it will be important to replicate these findings in a new set of participants. That is, participants who have highly left lateralized responses to language in fronto-temporal language regions are also more likely to have highly

left-lateralized responses to Python code in fronto-parietal networks. Together these data suggested that the language system does contribute to code comprehension.

Possible computational contributions of language network to code comprehension

We propose that during the comprehension of computer programs, information flows from visual cortex to the perisylvian language system, which initially constructs a language-like surface-level representation of the programming script. Then, the surface-level representation is transmitted to the logical reasoning system, where the algorithm is processed. This hypothesis is consistent with presence of multivariate information about programs in language areas and is supported by evidence for an earlier peak response to programs in language areas, relative to the fronto-parietal network.

According to this proposal, the "surface" representation of the language system describes the program in a non-computable format. By contrast, the fronto-parietal system actually simulates the program's execution, mapping between inputs and outputs, and changing the states of the variables throughout the execution of the program. When a input is provided, the fronto-parietal systems simulate the program to generate the output (Bunge, Kahn, Wallis, Miller, & Wagner, 2003; Crittenden, Mitchell, & Duncan, 2016; Pischedda, Görgen, Haynes, & Reverberi, 2017; Woolgar, Jackson, & Duncan, 2016; Woolgar, Thompson, Bor, & Duncan, 2011; Zhang, Kriegeskorte, Carlin, & Rowe, 2013). This interpretation of the fronto-parietal system's role is consistent with another finding reported by Srikant et al.(2022), where the fronto-parietal network encodes the "run-time behavior" of a program (e.g., the number of actual steps taken by the computer to execute the program) better than the language network.

Many questions remain regarding the precise role of the language network in code comprehension. Here, we discuss two possibilities, a "weak" recycling and a "strong" recycling hypothesis. According to the weak recycling idea, during code comprehension, the language network represents a natural language description of the algorithm represented in programming code, something like what a programmer writes in the comments section of programming script. Since humans are more familiar with natural than artificially designed programming languages, it is possible that a natural language description facilitates program comprehension. We call this a "weak" recycling hypothesis because the language network continues to perform its typical linguistic operations. Similar to using the language system to solve memorization-based math problems (6 times 6 is 36) (Maruyama, Pallier, Jobert, Sigman, & Dehaene, 2012).

Consistent with the weak recycling possibility, it appears to be easy for many programmers to generate language descriptions of their mental processes during coding. "Verbal protocols" are often used to study the cognitive basis of code processing (Hungerford, Hevner, & Collins, 2004; Lethbridge, Sim, & Singer, 2005; Letovsky, 1987; Letovsky & Soloway, 1986; Littman, Pinto, Letovsky, & Soloway, 1987; Pennington, 1987; Sharpe, 1997; von Mayrhauser & Vans, 1994). In such studies, programmers are instructed to "think aloud" during a code comprehension task and the linguistic utterances are analyzed to gain insight into how a programmer's understanding of a script evolves through time. For example, while reading an unfamiliar script, programmers may

begin by asking questions prompting closer inspection of other programming elements (e.g., “What does this other function ‘SRCH’ do?”, “Why are there 7 elements in the array ‘DBASE’?”), in a later stage, programmers’ utterances may become more affirmative, indicating a better understanding of the program (e.g., “So you access records by name in the data base.”, “Because they have that marker in there to determine if it’s deleted.”) (examples from (Letovsky, 1987)). One hypothesis is that such verbalization occurs covertly, even when there is no motor speech output. If such verbalizations are specific enough, they would account for distinctive neural patterns in the language network for different programming functions.

As opposed to the weak recycling view, the alternative “strong recycling” hypothesis is that the language network supports the representations of hierarchical “syntactic trees” of computer programs. On the strong recycling view, the language system’s capacity for representing hierarchical structures is “repurposed” to represent the structures found in computer code (Fitch et al., 2005; Friederici, Rueschemeyer, Hahne, & Fiebach, 2003; Humphries, Binder, Medler, & Liebenthal, 2006; Pallier, Devauchelle, & Dehaene, 2011). We call this view “strong recycling” because it would require modifying representations in the fronto-temporal language network to accommodate encoding the types of “trees” found in computer code. Such repurposing may even involve sub-specialization of some subset of the language network for representing programming code, akin to the development of the visual word form area (VWFA) in the ventral stream during literacy acquisition (Dehaene-Lambertz, Monzalvo, & Dehaene, 2018; Dehaene, Cohen, Morais, & Kolinsky, 2015; Dehaene, Le Clec'H, Poline, Le Bihan, & Cohen, 2002; Dehaene et al., 2010; McCandliss, Cohen, & Dehaene, 2003; Szwed, Vinckier, Cohen, & Dehaene, 2012). Potentially consistent with the idea that the language network represents the hierarchical trees of computer code, Srikant et al. (2022) found that the language network is sensitive to the number of nodes in the abstract syntactic tree of a program. However, it is also likely that more complex programs require more complex linguistic descriptions, so this data is also consistent with the weak recycling view.

At present, we favor the “weak” recycling view for several reasons. Given the similarity of natural languages to each other and their collective difference from programming languages, it seems likely that the language system is particularly suited for representing not just any “hierarchical tree” but specifically the types of trees that are found in natural languages. Even if the language network were capable of sufficient plasticity to accommodate trees found in computer code, coding is acquired well past the critical period of language acquisition. These considerations lead us to favor the “weak” recycling hypothesis, perhaps better termed the “reuse” for the relationship of language and code.

It is also worth noting that prior evidence suggests that other hierarchical cultural symbol systems such as formal logic, mathematics and music do not “recycle” the language network but instead rely on fronto-parietal circuits (Dehaene et al., 2022; Fedorenko & Varley, 2016; Monti, 2017; Monti et al., 2009, 2012). However, most of these studies used univariate approaches. It would be interesting to ask whether the contents of math and formal logic expressions could be decoded in language networks, despite low univariate response. If so, it might suggest that reuse of the language network is a wide-spread phenomenon across symbol systems.

Open questions and future directions

Many open questions remain regarding the role of the language system in code comprehension. One is whether the involvement of the language system generalizes to programming languages that are less language-like than Python, the programming language studied here. Specifically, does the language system represent program-relevant information only when the code is language-like? In the current study and Srikant et al. (2022), the programming language presented to participants was Python, which is among the more language-like high-level programming languages, in comparison with other major programming languages like C++ or Java. It is not known whether the language network encodes code-relevant information when the "code" does not consist of linguistic symbols.

A study by Ivanova et al. (2020) presented participants with ScratchJr programs in addition to Python programs. ScratchJr is a child-oriented graphic-based programming "language", where a program is "written" in a series of interlocked tiles containing icons and numbers. Whereas Ivanova et al. (2020) reported a slight preference for Python programs over lists of non-words in the language network, such an effect was not observed with ScratchJr "programs". However, Ivanova et al. (2020) did not test for multivariate decoding of ScratchJr programs in language regions. ScratchJr provides an interesting test-case to be used in future studies.

Another open question concerns the neural flow of information during program comprehension, akin to what has been processed for comprehension of natural languages (Price, 2010, 2012). Based on our findings, we propose that during comprehension of programs, information flows from visual regions to language circuits followed by fronto-parietal reasoning systems, with feedback from later to earlier regions throughout the process. However, it warrants further investigation whether the language system is behaviorally necessary, or merely active during code comprehension. It is possible that the logical reasoning system alone is sufficient to parse a program and construct the representations of the actual algorithms, whereas the surface-level representation constructed by the language network only facilitates the comprehension process. The necessity of the language network can be studied by conducting code comprehension experiments while inhibiting the language system using transcranial magnetic stimulation (TMS), or overloading the language system with verbal shadowing task, or with aphasic programmers.

Moving forward, as we delve deeper into understanding the role of the language system in code comprehension, it becomes evident that future investigations will not only illuminate the necessity of this system but also provide valuable insights into the mental processes involved in programming. Furthermore, these inquiries will help us unravel the intricate relationship between language and abstract symbolic reasoning, ultimately shedding light on the intriguing concept of neural recycling.

Reference

Bettencourt, K. C., & Xu, Y. (2016). Decoding the content of visual short-term memory under distraction in occipital and parietal areas. *Nature Neuroscience, 19*(1), 150-157.

Bunge, S. A., Kahn, I., Wallis, J. D., Miller, E. K., & Wagner, A. D. (2003). Neural circuits subserving the retrieval and maintenance of abstract rules. *Journal of neurophysiology, 90*(5), 3419-3428.

Bush, G., & Shin, L. M. (2006). The Multi-Source Interference Task: an fMRI task that reliably activates the cingulo-frontal-parietal cognitive/attention network. *Nature Protocols, 1*, 308. doi:10.1038/nprot.2006.48

Castelhano, J., Duarte, I. C., Ferreira, C., Duraes, J., Madeira, H., & Castelo-Branco, M. (2018). The role of the insula in intuitive expert bug detection in computer code: an fMRI study. *Brain imaging and behavior*. doi:10.1007/s11682-018-9885-1

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST), 2*(3), 1-27. doi:https://doi.org/10.1145/1961189.1961199

Crittenden, B., Mitchell, D., & Duncan, J. (2016). Task Encoding across the Multiple Demand Cortex Is Consistent with a Frontoparietal and Cingulo-Opercular Dual Networks Distinction. *The Journal of Neuroscience, 36*(23), 6147-6155. doi:10.1523/jneurosci.4590-15.2016

Dale, A. M., Fischl, B., & Sereno, M. I. (1999). Cortical surface-based analysis: I. Segmentation and surface reconstruction. *NeuroImage, 9*(2), 179-194. doi:https://doi.org/10.1006/nimg.1998.0395

Dehaene-Lambertz, G., Monzalvo, K., & Dehaene, S. (2018). The emergence of the visual word form: Longitudinal evolution of category-specific ventral visual areas during reading acquisition. *PLOS Biology, 16*(3), e2004103. doi:10.1371/journal.pbio.2004103

Dehaene, S., Al Roumi, F., Lakretz, Y., Planton, S., & Sablé-Meyer, M. (2022). Symbols and mental programs: a hypothesis about human singularity. *Trends in Cognitive Sciences*. doi:https://doi.org/10.1016/j.tics.2022.06.010

Dehaene, S., Cohen, L., Morais, J., & Kolinsky, R. (2015). Illiterate to literate: behavioural and cerebral changes induced by reading acquisition. *Nature Reviews Neuroscience, 16*(4), 234-244. doi:10.1038/nrn3924

Dehaene, S., Le Clec'H, G., Poline, J.-B., Le Bihan, D., & Cohen, L. (2002). The visual word form area: a prelexical representation of visual words in the fusiform gyrus. *Neuroreport, 13*(3), 321-325. doi:https://doi.org/10.1097/00001756-200203040-00015

Dehaene, S., Pegado, F., Braga, L. W., Ventura, P., Filho, G. N., Jobert, A., . . . Cohen, L. (2010). How Learning to Read Changes the Cortical Networks for Vision and Language. *Science, 330*(6009), 1359-1364. doi:https://doi.org/10.1126/science.1194140

Eklund, A., Knutsson, H., & Nichols, T. E. (2019). Cluster failure revisited: Impact of first level design and physiological noise on cluster false positive rates. *Human Brain Mapping, 40*(7), 2017-2032. doi:https://doi.org/10.1002/hbm.24350

Eklund, A., Nichols, T. E., & Knutsson, H. (2016). Cluster failure: Why fMRI inferences for spatial extent have inflated false-positive rates. *Proceedings of the National Academy of Sciences, 113*(28), 7900-7905. doi:https://doi.org/10.1073/pnas.1602413113

Elli, G. V., Lane, C., & Bedny, M. (2019). A Double Dissociation in Sensitivity to Verb and Noun Semantics Across Cortical Networks. *Cerebral Cortex*. doi:https://doi.org/10.1093/cercor/bhz014

Emrich, S. M., Riggall, A. C., LaRocque, J. J., & Postle, B. R. (2013). Distributed patterns of activity in sensory cortex reflect the precision of multiple items maintained in visual short-term memory. *Journal of Neuroscience, 33*(15), 6516-6523.

Ester, E. F., Serences, J. T., & Awh, E. (2009). Spatially global representations in human primary visual cortex during working memory maintenance. *Journal of Neuroscience, 29*(48), 15258-15265.
Etzel, J. A., Valchev, N., & Keysers, C. (2011). The impact of certain methodological choices on multivariate analysis of fMRI data with support vector machines. *NeuroImage, 54*(2), 1159-1167.
Fedorenko, E., Ivanova, A., Dhamala, R., & Bers, M. U. (2019). The language of programming: a cognitive perspective. *Trends in Cognitive Sciences*. doi:https://doi.org/10.1016/j.tics.2019.04.010
Fedorenko, E., & Varley, R. (2016). Language and thought are not the same thing: evidence from neuroimaging and neurological patients. *Annals of the New York Academy of Sciences, 1369*(1), 132-153. doi:https://doi.org/10.1111/nyas.13046
Fitch, W. T., Hauser, M. D., & Chomsky, N. (2005). The evolution of the language faculty: clarifications and implications. *Cognition, 97*(2), 179-210.
Floyd, B., Santander, T., & Weimer, W. (2017). *Decoding the representation of code in the brain: An fMRI study of code review and expertise.* Paper presented at the Proceedings of the 39th International Conference on Software Engineering.
Friederici, A., Chomsky, N., Berwick, R., Moro, A., & Bolhuis, J. (2017). Language, mind and brain. *Nature human behaviour, 1*(10), 713-722.
Friederici, A., Rueschemeyer, S.-A., Hahne, A., & Fiebach, C. J. (2003). The role of left inferior frontal and superior temporal cortex in sentence comprehension: localizing syntactic and semantic processes. *Cerebral Cortex, 13*(2), 170-177. doi:https://doi.org/10.1093/cercor/13.2.170
Glasser, M. F., Sotiropoulos, S. N., Wilson, J. A., Coalson, T. S., Fischl, B., Andersson, J. L., . . . Jenkinson, M. (2013). The minimal preprocessing pipelines for the Human Connectome Project. *NeuroImage, 80*, 105-124. doi:https://doi.org/10.1016/j.neuroimage.2013.04.127
Glezer, L. S., & Riesenhuber, M. (2013). Individual variability in location impacts orthographic selectivity in the “visual word form area”. *Journal of Neuroscience, 33*(27), 11221-11226.
Harrison, S. A., & Tong, F. (2009). Decoding reveals the contents of visual working memory in early visual areas. *Nature, 458*(7238), 632-635.
Hauser, M. D., Chomsky, N., & Fitch, W. T. (2002). The Faculty of Language: What Is It, Who Has It, and How Did It Evolve? *Science, 298*(5598), 1569-1579. doi:10.1126/science.298.5598.1569
Humphries, C., Binder, J., Medler, D., & Liebenthal, E. (2006). Syntactic and semantic modulation of neural activity during auditory sentence comprehension. *Cognitive Neuroscience, Journal of, 18*(4), 665-679.
Hungerford, B. C., Hevner, A. R., & Collins, R. W. (2004). Reviewing software diagrams: A cognitive study. *IEEE Transactions on Software Engineering, 30*(2), 82-96.
Ikutani, Y., Kubo, T., Nishida, S., Hata, H., Matsumoto, K., Ikeda, K., & Nishimoto, S. (2021). Expert Programmers Have Fine-Tuned Cortical Representations of Source Code. *eneuro, 8*(1), ENEURO.0405-0420.2020. doi:10.1523/ENEURO.0405-20.2020
Ivanova, A. A., Srikant, S., Sueoka, Y., Kean, H. H., Dhamala, R., O'Reilly, U.-M., . . . Fedorenko, E. (2020). Comprehension of computer code relies primarily on domain-general executive brain regions. *eLife, 9*, e58906. doi:10.7554/eLife.58906
Kanjlia, S., Lane, C., Feigenson, L., & Bedny, M. (2016). Absence of visual experience modifies the neural basis of numerical thinking. *Proceedings of the National Academy of Sciences, 113*(40), 11172-11177.
Kim, J. S., Kanjlia, S., Merabet, L. B., & Bedny, M. (2017). Development of the visual word form area requires visual experience: Evidence from blind Braille readers. *Journal of Neuroscience, 37*(47), 11495-11504. doi:https://doi.org/10.1523/jneurosci.0997-17.2017

Kriegeskorte, N., Goebel, R., & Bandettini, P. (2006). Information-based functional brain mapping. *Proceedings of the National Academy of Sciences of the United States of America, 103*(10), 3863-3868. doi:https://doi.org/10.1073/pnas.0600244103

Kriegeskorte, N., Simmons, W. K., Bellgowan, P. S., & Baker, C. I. (2009). Circular analysis in systems neuroscience: the dangers of double dipping. *Nature Neuroscience, 12*(5), 535-540.

Krueger, R., Huang, Y., Liu, X., Santander, T., Weimer, W., & Leach, K. (2020). *Neurological Divide: An fMRI Study of Prose and Code Writing.* Paper presented at the 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE).

Lee, S., & Kable, J. W. (2018). Simple but robust improvement in multivoxel pattern classification. *PLoS ONE, 13*(11), e0207083.

Lethbridge, T. C., Sim, S. E., & Singer, J. (2005). Studying software engineers: Data collection techniques for software field studies. *Empirical Software Engineering, 10*, 311-341.

Letovsky, S. (1987). Cognitive processes in program comprehension. *Journal of Systems and Software, 7*(4), 325-339. doi:https://doi.org/10.1016/0164-1212(87)90032-X

Letovsky, S., & Soloway, E. (1986). Delocalized plans and program comprehension. *IEEE software, 3*(3), 41.

Littman, D. C., Pinto, J., Letovsky, S., & Soloway, E. (1987). Mental models and software maintenance. *Journal of Systems and Software, 7*(4), 341-355. doi:https://doi.org/10.1016/0164-1212(87)90033-1

Liu, Y.-F., Kim, J., Wilson, C., & Bedny, M. (2020). Computer code comprehension shares neural resources with formal logical inference in the fronto-parietal network. *eLife, 9*, e59340.

Maruyama, M., Pallier, C., Jobert, A., Sigman, M., & Dehaene, S. (2012). The cortical representation of simple mathematical expressions. *NeuroImage, 61*(4), 1444-1460. doi:https://doi.org/10.1016/j.neuroimage.2012.04.020

McCandliss, B. D., Cohen, L., & Dehaene, S. (2003). The visual word form area: expertise for reading in the fusiform gyrus. *Trends in Cognitive Sciences, 7*(7), 293-299. doi:https://doi.org/10.1016/S1364-6613(03)00134-7

McCoy, L. P., & Burton, J. K. (1988). The relationship of computer programming and mathematics in secondary students.

Monti, M. (2017). The role of language in structure-dependent cognition *Neural mechanisms of language* (pp. 81-101): Springer.

Monti, M., & Osherson, D. (2012). Logic, language and the brain. *Brain Research, 1428*, 33-42. doi:https://doi.org/10.1016/j.brainres.2011.05.061

Monti, M., Parsons, L., & Osherson, D. (2009). The boundaries of language and thought in deductive inference. *Proceedings of the National Academy of Sciences, 106*(30), 12554-12559.

Monti, M., Parsons, L., & Osherson, D. (2012). Thought beyond language: neural dissociation of algebra and natural language. *Psychological Science, 23*(8), 914-922. doi:10.1177/0956797612437427

Mumford, J. A., Davis, T., & Poldrack, R. A. (2014). The impact of study design on pattern estimation for single-trial multivariate pattern analysis. *NeuroImage, 103*, 130-138.

Musz, E., Loiotile, R., Chen, J., & Bedny, M. (2022). Naturalistic Audio-Movies reveal common spatial organization across “visual” cortices of different blind individuals. *Cerebral Cortex*, bhac048. doi:10.1093/cercor/bhac048

Nieto-Castañón, A., & Fedorenko, E. (2012). Subject-specific functional localizers increase sensitivity and functional resolution of multi-subject analyses. *NeuroImage, 63*(3), 1646-1669.

Pallier, C., Devauchelle, A.-D., & Dehaene, S. (2011). Cortical representation of the constituent structure of sentences. *Proceedings of the National Academy of Sciences*. doi:10.1073/pnas.1018711108

Pandža, N. B. (2016). Computer programming as a second language *Advances in Human Factors in Cybersecurity* (pp. 439-445): Springer, Cham.
Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New ideas in psychology, 2*(2), 137-168.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *Journal of machine Learning research, 12*(Oct), 2825-2830. doi:https://dl.acm.org/doi/10.5555/1953048.2078195
Peirce, J., Gray, J. R., Simpson, S., MacAskill, M., Höchenberger, R., Sogo, H., . . . Lindeløv, J. K. (2019). PsychoPy2: Experiments in behavior made easy. *Behav Res Methods, 51*(1), 195-203. doi:10.3758/s13428-018-01193-y
Peitek, N., Siegmund, J., Apel, S., Kästner, C., Parnin, C., Bethmann, A., . . . Brechmann, A. (2018). A look into programmers' heads. *IEEE Transactions on Software Engineering*, 1-1. doi:10.1109/TSE.2018.2863303
Pennington, N. (1987). Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive Psychology, 19*(3), 295-341. doi:https://doi.org/10.1016/0010-0285(87)90007-7
Pischedda, D., Görgen, K., Haynes, J.-D., & Reverberi, C. (2017). Neural Representations of Hierarchical Rule Sets: The Human Control System Represents Rules Irrespective of the Hierarchical Level to Which They Belong. *The Journal of Neuroscience, 37*(50), 12281-12296. doi:10.1523/jneurosci.3088-16.2017
Portnoff, S. R. (2018). The introductory computer programming course is first and foremost a language course. *ACM Inroads, 9*(2), 34-52.
Prat, C. S., Madhyastha, T. M., Mottarella, M. J., & Kuo, C.-H. (2020). Relating natural language aptitude to individual differences in learning programming languages. *Scientific reports, 10*(1), 3817. doi:10.1038/s41598-020-60661-8
Price, C. (2010). The anatomy of language: a review of 100 fMRI studies published in 2009. *Annals of the New York Academy of Sciences, 1191*(1), 62-88. doi:https://doi.org/10.1111/j.1749-6632.2010.05444.x
Price, C. (2012). A review and synthesis of the first 20 years of PET and fMRI studies of heard speech, spoken language and reading. *NeuroImage, 62*(2), 816-847.
Regev, M., Honey, C., Simony, E., & Hasson, U. (2013). Selective and Invariant Neural Responses to Spoken and Written Narratives. *Journal of Neuroscience, 33*(40), 15978-15988. doi:10.1523/jneurosci.1580-13.2013
Riggall, A. C., & Postle, B. R. (2012). The relationship between working memory storage and elevated activity as measured with functional magnetic resonance imaging. *Journal of Neuroscience, 32*(38), 12990-12998.
Schaefer, A., Kong, R., Gordon, E. M., Laumann, T. O., Zuo, X.-N., Holmes, A. J., . . . Yeo, B. T. T. (2018). Local-global parcellation of the human cerebral cortex from intrinsic functional connectivity MRI. *Cerebral Cortex, 28*(9), 3095-3114. doi:10.1093/cercor/bhx179
Schreiber, K., & Krekelberg, B. (2013). The Statistical Analysis of Multi-Voxel Patterns in Functional Imaging. *PLoS ONE, 8*(7), e69328. doi:https://doi.org/10.1371/journal.pone.0069328
Sharpe, S. (1997). Unifying Theories of Program Comprehension. *Journal of Computer Information Systems, 38*(1), 86-93. doi:10.1080/08874417.1997.11647312
Shute, V. J. (1991). Who is Likely to Acquire Programming Skills? *Journal of Educational Computing Research, 7*(1), 1-24. doi:10.2190/VQJD-T1YD-5WVB-RYPJ
Siegmund, J., Kästner, C., Apel, S., Parnin, C., Bethmann, A., Leich, T., . . . Brechmann, A. (2014). *Understanding understanding source code with functional magnetic resonance imaging.* Paper presented at the Proceedings of the 36th International Conference on Software Engineering.

Smith, S. M., Jenkinson, M., Woolrich, M. W., Beckmann, C. F., Behrens, T. E., Johansen-Berg, H., . . . Flitney, D. E. (2004). Advances in functional and structural MR image analysis and implementation as FSL. *NeuroImage, 23*, S208-S219. doi:https://doi.org/10.1016/j.neuroimage.2004.07.051

Srikant, S., Lipkin, B., Ivanova, A. A., Fedorenko, E., & O'Reilly, U.-M. (2022). *Convergent Representations of Computer Programs in Human and Artificial Neural Networks*. Paper presented at the 36th Conference on Neural Information Processing Systems.

Stehr, D. A., Garcia, J. O., Pyles, J. A., & Grossman, E. D. (2023). Optimizing multivariate pattern classification in rapid event-related designs. *Journal of neuroscience methods, 387*, 109808.

Stelzer, J., Chen, Y., & Turner, R. (2013). Statistical inference and multiple testing correction in classification-based multi-voxel pattern analysis (MVPA): Random permutations and cluster size control. *NeuroImage, 65*, 69-82. doi:https://doi.org/10.1016/j.neuroimage.2012.09.063

Su, L., Fonteneau, E., Marslen-Wilson, W., & Kriegeskorte, N. (2012, 2-4 July 2012). *Spatiotemporal Searchlight Representational Similarity Analysis in EMEG Source Space.* Paper presented at the 2012 Second International Workshop on Pattern Recognition in NeuroImaging.

Szwed, M., Vinckier, F., Cohen, L., & Dehaene, S. (2012). Towards a universal neurobiological architecture for learning to read. *Behavioral and Brain Sciences, 35*(5), 308-309.

Valente, G., Castellanos, A. L., Hausfeld, L., De Martino, F., & Formisano, E. (2021). Cross-validation and permutations in MVPA: Validity of permutation strategies and power of cross-validation schemes. *NeuroImage, 238*, 118145.

von Mayrhauser, A., & Vans, A. M. (1994). *Comprehension processes during large scale maintenance.* Paper presented at the Proceedings of 16th International Conference on Software Engineering.

Winkler, A. M., Ridgway, G. R., Webster, M. A., Smith, S. M., & Nichols, T. E. (2014). Permutation inference for the general linear model. *NeuroImage, 92*(100), 381-397. doi:https://doi.org/10.1016/j.neuroimage.2014.01.060

Woolgar, A., Jackson, J., & Duncan, J. (2016). Coding of Visual, Auditory, Rule, and Response Information in the Brain: 10 Years of Multivoxel Pattern Analysis. *Journal of Cognitive Neuroscience, 28*(10), 1433-1454. doi:10.1162/jocn_a_00981

Woolgar, A., Thompson, R., Bor, D., & Duncan, J. (2011). Multi-voxel coding of stimuli, rules, and responses in human frontoparietal cortex. *NeuroImage, 56*(2), 744-752. doi:https://doi.org/10.1016/j.neuroimage.2010.04.035

Xu, S., Li, Y., & Liu, J. (2021). The Neural Correlates of Computational Thinking: Collaboration of Distinct Cognitive Components Revealed by fMRI. *Cerebral Cortex*. doi:10.1093/cercor/bhab182

Yang, C., Crain, S., Berwick, R. C., Chomsky, N., & Bolhuis, J. J. (2017). The growth of language: Universal Grammar, experience, and principles of computation. *Neuroscience & Biobehavioral Reviews, 81*, 103-119.

Zhang, J., Kriegeskorte, N., Carlin, J. D., & Rowe, J. B. (2013). Choosing the rules: distinct and overlapping frontoparietal representations of task rules for perceptual decisions. *The Journal of Neuroscience, 33*(29), 11852-11862. doi:10.1523/jneurosci.5193-12.2013

Table 1. Clusters revealed by the searchlight MVPA decoding. FWER corrected. Only clusters with more than 50 vertices are included.

	peak MNI coordinates			Cluster size		peak-p
	X	Y	Z	vertices	mm2	
Left hemisphere						
Superior temporal sulcus/angular gyrus/intraparietal sulcus	-46.6	-69.2	21.9	3169	5260.58	3.62E-09
Superior frontal sulcus/middle frontal gyrus/precentral gyrus & sulcus	-24.9	16.8	40	979	2022.29	8.55E-09
Precuneus	-6.3	-59	32	526	913.58	1.23E-07
Superior frontal gyrus	-16.4	18.8	59	395	1026.33	3.27E-08
Inferior frontal gyrus	-52.7	14.1	5.1	213	510.77	1.97E-07
Posterior dorsal cingulate gyrus	-7.4	-40.8	39.1	121	260.38	1.31E-07
Superior frontal gyrus	-13.4	53.1	32.4	89	199.66	2.42E-06
Middle frontal gyrus	-45.4	30.6	25.8	86	150.12	3.44E-05
Medial occipto-temporal sulcus	-32.6	-48.9	-10.2	74	173.29	4.02E-07
Orbital gyrus	-41.2	26.7	-16.1	71	172.65	7.33E-07
Lingual gyrus	-3.8	-91.1	-6.8	51	136.47	7.01E-06
Right hemisphere						
Intraparietal sulcus/superior temporal sulcus/angular gyrus	35.5	-54.8	39.8	1036	1954.92	9.23E-08
Superior frontal sulcus	27	13.1	45.5	315	551.61	3.07E-07
Precuneus	5.9	-68.1	47.6	266	504.43	1.52E-06
Middle frontal gyrus/inferior frontal sulcus	47.7	23.4	31.5	193	342.6	4.56E-07
Precuneus	7.3	-54.9	47.1	87	113.06	6.09E-06
Superior frontal gyrus	11.5	54.7	31.8	60	205.07	3.13E-06
Superior frontal gyrus	16.5	38.8	45	57	162.05	6.77E-06
Lingual gyrus	3.1	-79.2	0.2	52	195.12	4.26E-06

Supplementary Table 1. In either regions of interest (ROI) in the fronto-parietal reasoning network (the intraparietal sulcus IPS and the lateral prefrontal cortex PFC), decoding accuracy is compared between two methods for the definition of the ROI search space. In our current study, the ROI search spaces were defined using the logic>sentence contrast. An alternative was to define them based on the real code > scrambled fake code contrast. However, the decoding accuracy derived from the two methods are highly correlated and statistically indistinguishable.

	IPS	**PFC**
Logic>sentence contrast	68.9% (SD=7.17%)	64.4% (SD=7.82%)
Real>fake contrast	67.8% (SD=8.24%)	64.8% (SD=7.65%)
Pearson correlation	r=0.88, p<0.001	r=0.96, p<0.001
Paired t-test	t=1.05, p=0.31	t=-0.75, p=0.47

	FOR	IF

Real	def fun(input):
	    rerult = [“input”]
	    for ii in input.split(“_”):
	        result += [ii[0].lower()]
	    return result
		def fun(input):
		    result = [“result”]
		    if len(input.split(“_”))<3:
		        result += input.split(“_”)
		    return result

Fake	input :def)fun(
	    result=input” “[ ]
	    input” in) split.for” ii(:_
	        ii] result[lower=0.()] +[
	    result return
		input :def)fun(
		    result=result” “[ ]
		    split(input)if.3 “len)(:_”<
		        result.split” input”_)(+ =
		    result return

Figure 1. Example stimuli. The top row shows a real FOR function and a real IF function. The bottom row shows a scrambled (fake) FOR function and a fake IF function. Each fake function was created by scrambling the words and symbols in each line of the corresponding real function. During the experiment, all stimuli were presented on a black background. Real functions were presented in a white font, and fake functions were presented in a yellow font as a visual reminder to prevent participants from temporarily thinking the fake functions were real functions. For an illustration for the experiment design, please refer to Liu et al. (2020).

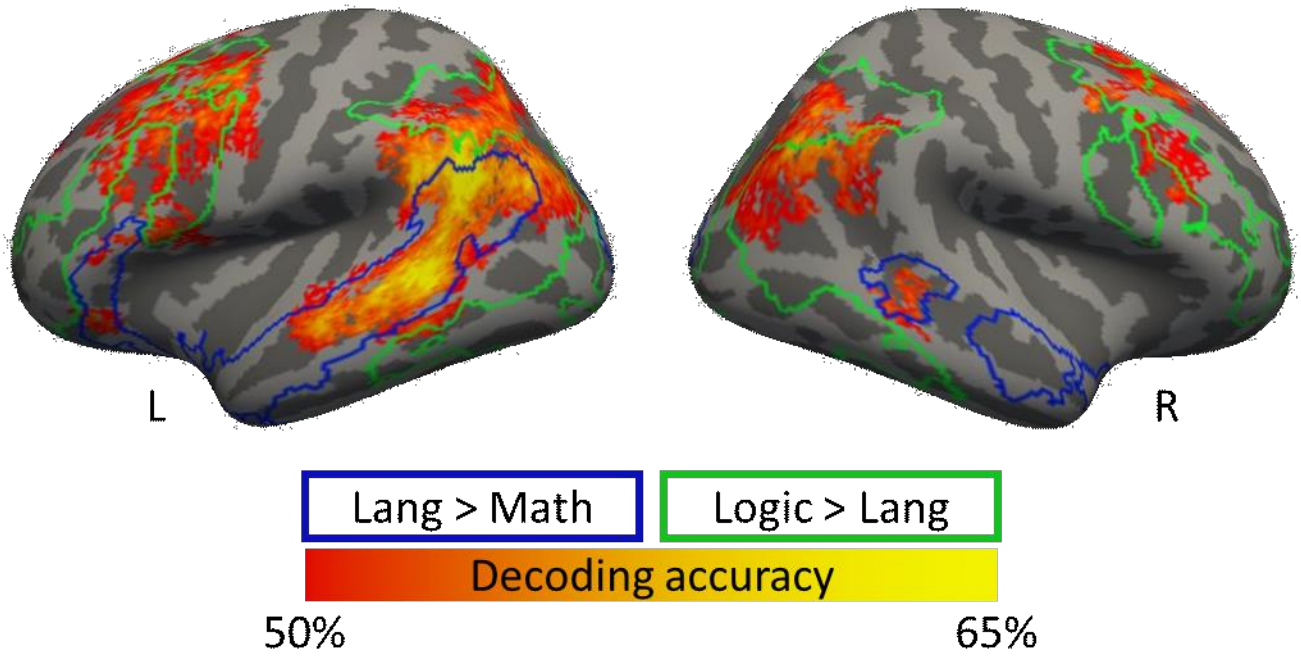


Figure 2. Searchlight multivariate pattern analysis (MVPA) FOR-vs-IF Python function decoding accuracy map. Family-wise error rate (FWER) was controlled by applying cluster-based permutation correction, with a vertex-wise cluster-forming threshold of uncorrected $p<0.001$, and a cluster-wise threshold of $p<0.05$. The blue outlines denote the language-responsive network defined based on the sentence>math group contrast derived from the localizer scan, whereas the green outlines denote the logic-responsive network defined based on the logic>sentence group contrast. All the vertices shown in this figure have significantly above-chance accuracy. For the list of clusters which passed the FWER correction, please see Table 1.

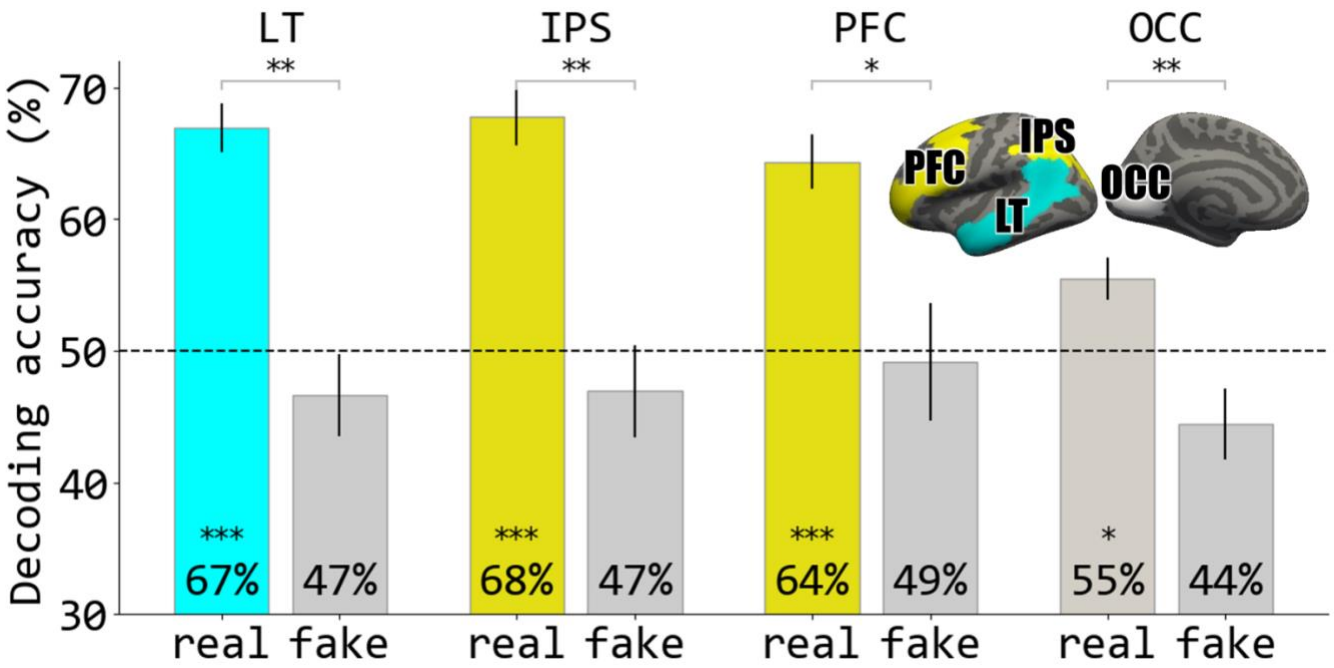


Figure 3. FOR-vs-IF Python function MVPA decoding accuracy in language-responsive left lateral temporal cortex (LT), code-responsive left intraparietal sulcus (IPS), code-responsive left lateral prefrontal cortex (PFC), and left primary visual medial occipital cortex (OCC). The “real” is the accuracy of decoding if vs. for Python functions. “Fake” is the accuracy of fake/scrambled function decoding. The search spaces are delineated on the inset brain map. Chance level is 50%. Error bars denote standard error of the decoding accuracy. *p<0.05, **p<0.01, ***p<0.001

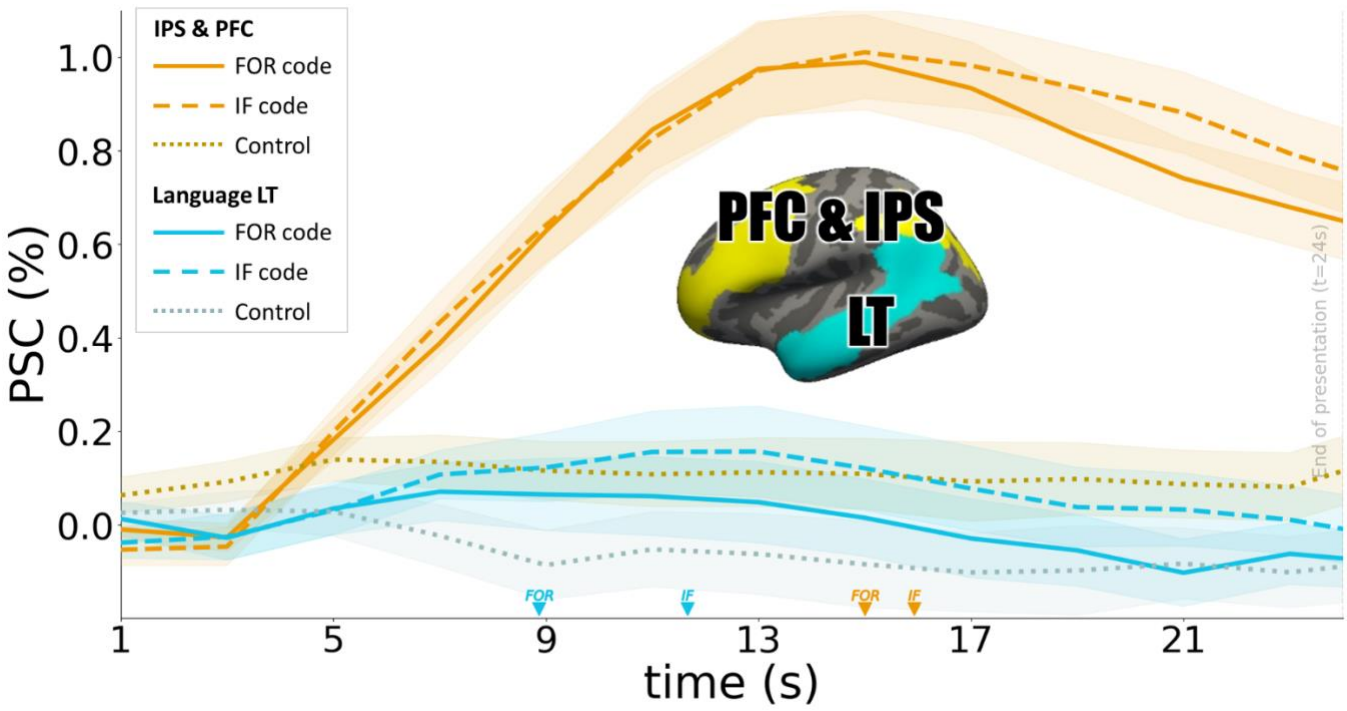
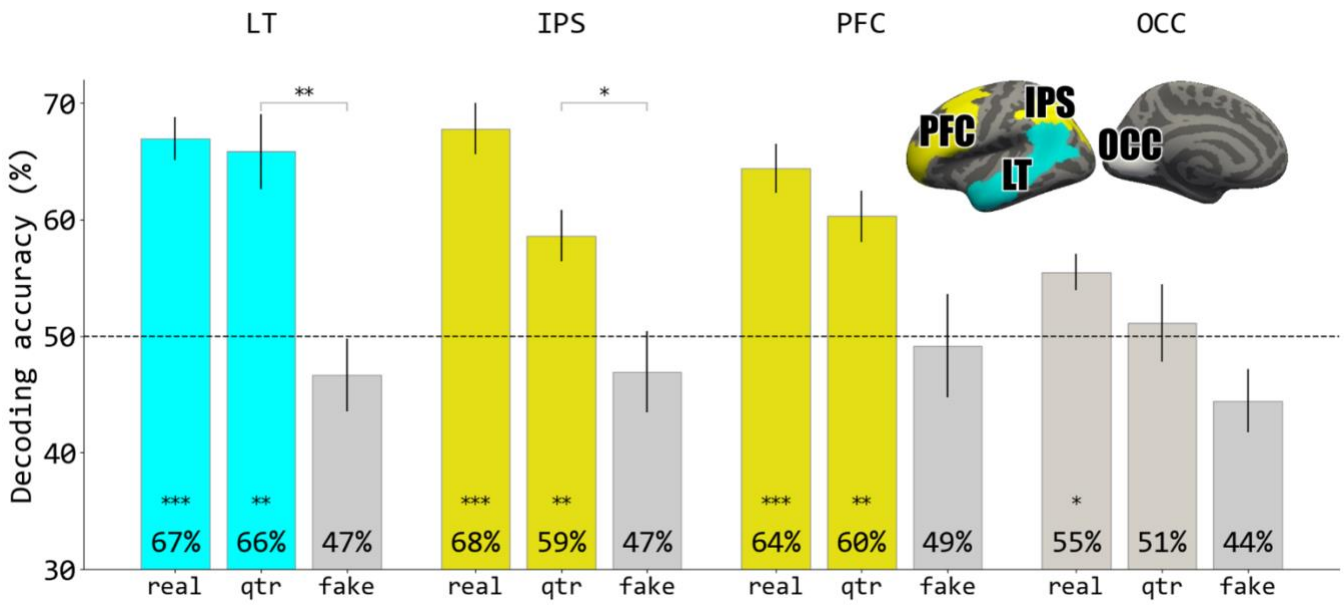
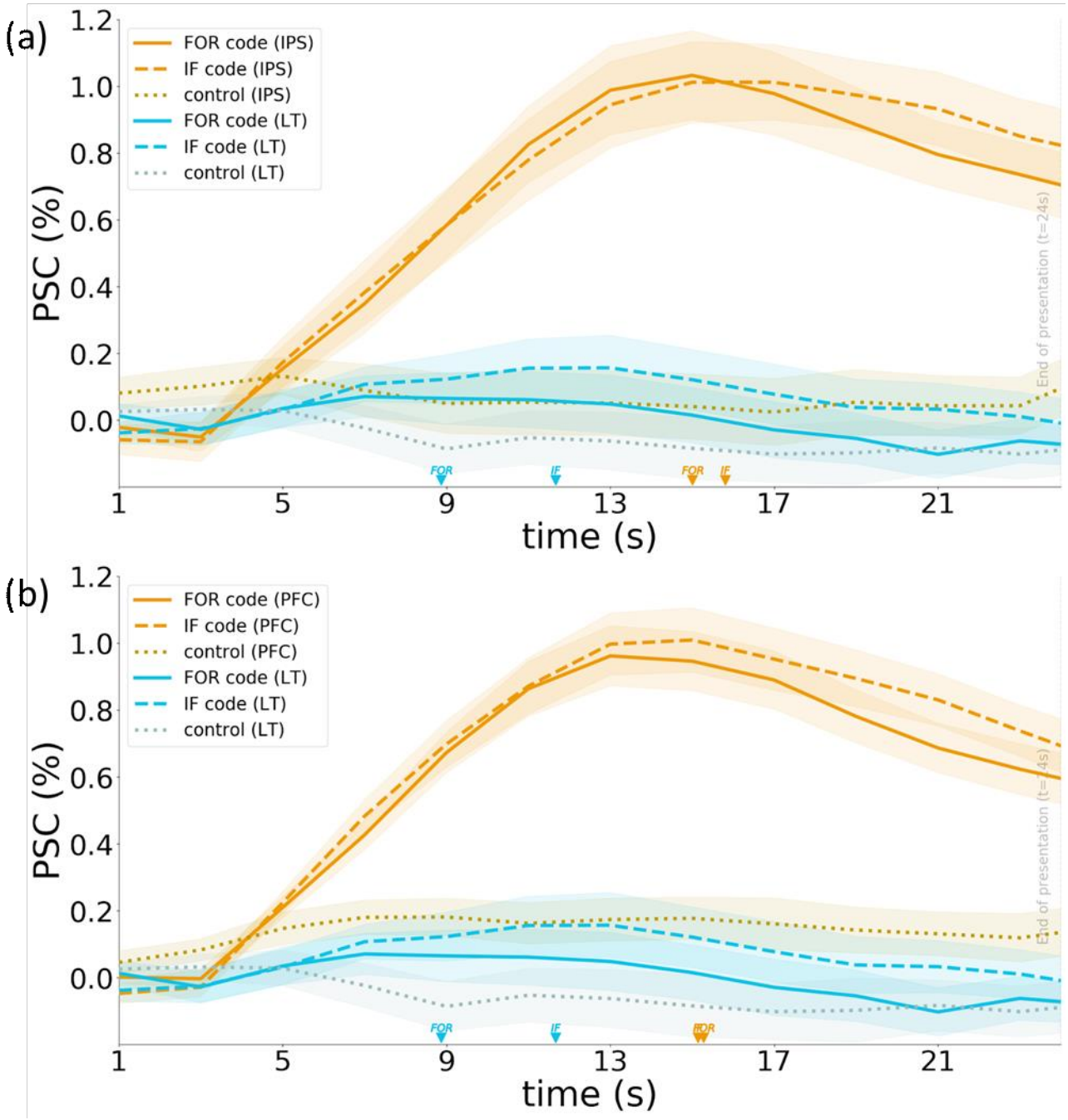


Figure 4. Percent signal change (PSC) time courses of different conditions, averaged across participants: FOR (solid line), IF (dashed line), and scrambled fake function (dotted line). Blue lines: PSC time courses extracted from the top 5% language-responsive vertices in the left lateral temporal (LT) search space. The search spaces are delineated on the inset brain map. Yellow lines: PSC time courses extracted from the top 5% code-responsive vertices (using leave-one-run-out method) in the left intraparietal sulcus (IPS) and the lateral prefrontal cortex (PFC), averaged across IPS and PFC. Translucent shades denote standard error. The across-participant average peak time of the responses to FOR and IF functions (Blue: LT, Yellow: IPS&PFC) are denoted along the X axis.



Supplementary Figure 1. FOR-vs-IF Python function MVPA decoding accuracy in the following regions of interest (ROIs) in the left hemisphere: language-responsive lateral temporal cortex (LT), code-responsive intraparietal sulcus (IPS), code-responsive lateral prefrontal cortex (PFC), and primary visual medial occipital cortex (OCC). In each group of bars, the left one ("real") is the accuracy of real function decoding, the middle one ("qtr") is the accuracy we observed when only a quarter of the real function data were used, and the right one ("fake") is the accuracy of fake function decoding. In order not to complicate the figure, the significance of the differences between "real" and "fake" bars in the LT, the IPS, and the PFC are not denoted in this figure (this information is included in Figure 3). The search spaces are delineated on the inset brain map. Chance level is 50%. Error bars denote standard error of the decoding accuracy. *p<0.05, **p<0.01, ***p<0.001



Supplementary Figure 2: Comparison of percent signal change time courses between (a) the IPS and the LT (b) the PFC and the LT. IPS: left intraparietal sulcus. PFC: left lateral prefrontal cortex. LT: left lateral temporal cortex. The figures are identical to Figure 4, except for "IPS&PFC" being replaced with either IPS or PFC. Responses in the LT was faster than in the IPS (FOR: mean peak time = 8.87s in the LT, SD=6.39s; mean = 15s in the IPS, SD=3.86s; $t(14)=-4.61$, $p<0.001$. IF: mean peak time = 11.67s in the LT, SD=6.14s; mean = 15.8s in the IPS, SD=4.37s; $t(14)=-3.72$, $p<0.005$). Responses in the LT was also faster than in the PFC (FOR: mean peak time = 8.87s in the LT, SD=6.39s; mean = 15.26s in the PFC, SD=3.57s;

t(14)=-4.82, p<0.001. IF: mean peak time = 11.67s in the LT, SD=6.14s; mean = 15.13s in the PFC, SD=3.96s; t(14)=-3.45, p<0.005)